In the Claims:

1. (Original) In a system-level design and integration system which facilitates composing behavior, capturing architecture and mapping behavior onto architecture, a method for specification of scheduling for simulation and implementation of consumer embedded systems; the method comprising the steps of:

a) identifying the schedulers in an architecture;

b) identifying the schedulables constituting behavior;

c) assigning schedulables and schedulers to each other;

d) arranging for the schedulers to find their respective assigned schedulables;

e) arranging for the schedulables to find their respective assigned scheduler;

f) sending an event to a schedulable;

g) sending an activation notice from the schedulable in step f) to its assigned scheduler;

h) sending a message from the assigned scheduler of step g) to the schedulable of step f) to start its behavior's reaction; and

I) sending a finish notice from the schedulable of step f) to the scheduler of step g) when said reaction is completed.

2. (Original) The method recited in claim 1 further comprising the steps of:

h1) having the scheduler of step g) determine whether the reaction of the schedulable of step f) must be preempted before it is finished;

h2) if in step h1) preemption is required, sending a suspend message from the scheduler of step g) to the schedulable of step f) to temporarily halt the execution of its reaction;

h3) if in step h1) preemption is required, causing the schedulable of step f) to halt its reaction in response to the suspend message;

h4) if in step H1) preemption is required, eventually sending a resume message from the scheduler of step g) to the schedulable of step f).

3. (Original) The method recited in claim 2 further comprising the steps of:

providing a next function which occurs when a scheduler receives an activation notice from an assigned schedulable while said scheduler is idle and when an assigned schedulable has finished a reaction;

said next function sending a handle to a schedulable which will be starting its behavior's reaction next.

4. (Original) The method recited in claim 2 further comprising the steps of:

providing a "preempt" function which occurs when an activation notice is received by a scheduler while its assigned schedulable is already reacting;

said "preempt" function determining whether a preemption should occur and whether such a preemption should be accomplished by suspending or aborting the reaction to be preempted.

5. (Original) The method recited in claim 2 further comprising the steps of:

providing a "next" function which occurs when a scheduler receives an activation notice from an assigned schedulable while said scheduler is idle and when an assigned schedulable has finished a reaction;

said next function sending a handle to a schedulable which will be starting its behavior's reaction next;

providing a preempt function which occurs when an activation notice is received by a scheduler while its assigned schedulable is already reacting;

said preempt function determining whether a preemption should occur and whether such a preemption should be accomplished by suspending or aborting the reaction to be preempted.

6. (Original) The method recited in claim 5 further comprising the steps of:

assigning transition times for completing selected states in said scheduler and schedulable operations including the steps of starting and resuming reactions, finishing reactions, and suspending and aborting reactions.

7. (Original) The method recited in claim 1 further comprising the step of:

scheduling behaviors of said schedulables based in part on the priorities of mapping assignments of said schedulables.

8. (Original) The method recited in claim 2 wherein steps h1) to h4) are carried out in accordance with a cyclo-static scheduling policy.

9. (Original) The method recited in claim 2 wherein steps h1) to h4) are carried out in accordance with a non-preemptive static priority scheduling policy.

10. (Original) The method recited in claim 2 wherein steps h1) to h4) are carried out in accordance with a preemptive static priority scheduling policy.

11. (Original) The method recited in claim 2 wherein steps h1) to h4) are carried out in accordance with a selected scheduling policy.

12. (Original) The method recited in claim 11 further comprising the step of altering said selected scheduling policy based upon behavior of said embedded systems.

13. (Original) The method recited in claim 11 further comprising the step of generating software code for a consumer device to implement said selected scheduling policy.

14. (Original) The method recited in claim 11 further comprising the step of implementing said scheduling policy by interfacing to existing commercially available software.

15. (Original) A method for generating scheduling implementation for an embedded system expressed as an architecture, a behavior and a mapping of the behavior into the architecture; the method comprising the steps of:

a) identifying the schedulers in an architecture;

b) identifying the schedulables constituting behavior;

c) assigning schedulables and schedulers to each other;

d) arranging for the schedulers to find their respective assigned schedulables;

e) arranging for the schedulables to find their respective assigned scheduler;

f) sending an event to a schedulable;

g) sending an activation notice from the schedulable in step f) to its assigned scheduler;

h) sending a message from the assigned scheduler of step g) to the schedulable of step f) to start its behavior's reaction; and

i) sending a finish notice from the schedulable of step f) to the scheduler of step g) when said reaction is completed.

16. (Original) The method recited in claim 15 further comprising the steps of:

h1) having the scheduler of step g) determine whether the reaction of the schedulable of step f) must be preempted before it is finished;

h2) if in step h1) preemption is required, sending a suspend message from the scheduler of step g) to the schedulable of step f) to temporarily halt the execution of its reaction;

h3) if in step h1) preemption is required, causing the schedulable of step f) to halt its reaction in response to the suspend message;

h4) if in step H1) preemption is required, eventually sending a resume message from the scheduler of step g) to the schedulable of step f).

17. (Original) The method recited in claim 16 further comprising the steps of:

providing a next function which occurs when a scheduler receives an activation notice from an assigned schedulable while said scheduler is idle and when an assigned schedulable has finished a reaction;

said next function sending a handle to a schedulable which will be starting its behavior's reaction next.

18. (Original) The method recited in claim 16 further comprising the steps of:

providing a preempt function which occurs when an activation notice is received by a scheduler while its assigned schedulable is already reacting;

said preempt function determining whether a preemption should occur and whether such a preemption should be accomplished by suspending or aborting the reaction to be preempted.

19. (Original) The method recited in claim 16 further comprising the steps of:

providing a next function which occurs when a scheduler receives an activation notice from an assigned schedulable while said scheduler is idle and when an assigned schedulable has finished a reaction;

said next function sending a handle to a schedulable which will be starting its behavior's reaction next.

providing a preempt function which occurs when an activation notice is received by a scheduler while its assigned schedulable is already reacting;

said preempt function determining whether a preemption should occur and whether such a preemption should be accomplished by suspending or aborting the reaction to be preempted.

20. (Original) The method recited in claim 19 further comprising the steps of:

assigning transition times for completing selected states in said scheduler and schedulable operations, including the steps of starting and resuming reactions, finishing reactions, and suspending and aborting reactions.

21. (Original) The method recited in claim 15 further comprising the step of:

scheduling behaviors of said schedulables based in part on the priorities of mapping assignments of said schedulables.

22. (Original) The method recited in claim 16 wherein steps h1) to h4) are carried out in accordance with a cyclo-static scheduling policy.

23. (Original) The method recited in claim 16 wherein steps h1) to h4) are carried out in accordance with a cyclo-static scheduling policy.

24. (Original) The method recited in claim 16 wherein steps h1) to h4) are carried out in accordance with a preemptive static priority scheduling policy.

25. (Original) The method recited in claim 16 wherein steps h1) to h4) are carried out in accordance with a selected scheduling policy.

26. (Original) The method recited in claim 25 further comprising the step of altering said selected scheduling policy based upon behavior of said embedded systems.

27. (Original) The method recited in claim 25 further comprising the step of generating software code for a consumer device to implement said selected scheduling policy.

28. (Original) The method recited in claim 25 further comprising the step of implementing said scheduling policy by interfacing to existing commercially available software.

29. (New) In a system-level design and integration system which facilitates composing behavior, capturing architecture and mapping behavior onto architecture, a method for specification of scheduling for simulation and implementation of consumer embedded systems; the method comprising the steps of:

a) identifying the schedulers in an architecture;

b) identifying the schedulables constituting behavior;

c) assigning schedulables and schedulers to each other;

d) arranging for the schedulers to find their respective assigned schedulables;

e) arranging for the schedulables to find their respective assigned scheduler;

f) sending an event to a schedulable;

g) sending an activation notice from the schedulable in step f) to its assigned scheduler;

h) sending a message from the assigned scheduler of step g) to the schedulable of step f) to start its behavior's reaction;

h1) having the scheduler of step g) determine whether the reaction of the schedulable of step f) must be preempted before it is finished;

h2) if in step h1) preemption is required, sending a suspend message from the scheduler of step g) to the schedulable of step f) to temporarily halt the execution of its reaction;

h3) if in step h1) preemption is required, causing the schedulable of step f) to halt its reaction in response to the suspend message;

h4) if in step h1) preemption is required, eventually sending a resume message from the scheduler of step g) to the schedulable of step f); and

i) sending a finish notice from the schedulable of step f) to the scheduler of step g) when said reaction is completed.

30. (New) The method according to Claim 29, wherein

each message comprises one in a series of schedulable-scheduler messages comprising, from a set of messages: (1) an activation message from the schedulable to a scheduler indicating the schedulable wants to react to an event; (2) a start messsage from the scheduler instructing the schedulable to start; (3) a suspend message sent form the scheduler to suspend the schedulable's reaction; (4) a resume message from the scheduler to the schedulable; and (5) a completion message from the schedulable to the scheduler.

31. (New) The method according to Claim 30, wherein each message of each scheduler is modeled based on a stored value of how long it takes to transition through states specified by the message.

32. (New) The method according to Claim 29, wherein at least one of the schedulables is a hierarchy of lesser schedulables.

33. (New) The method according to Claim 30, wherein the schedulables react to events, and the events comprise a set of events having a timestamp that indicates an order in which a simulator processes the events.

34. (New) The method according to Claim 33, wherein the order of processing comprises the smallest time stamped event first.

35. (New) The method according to Claim 29, wherein resources utilized to process the schedulables are architectural resources allocated to the schedulables via a contention process.

36. (New) The method according to Claim 29, wherein the schedulables are not assigned at a fixed frequency.

37. (New) The method according to Claim 29, wherein the schedulables are activated based on a combination of activation of the schedulable and a priority assigned to the schedulable.

38. (New) The method according to Claim 29, wherein at least one of the schedulers is configured to implement a scheduler interface.

39. (New) The method according to Claim 38, wherein the scheduler interface is part of a hierarchy of schedulers, such that the scheduler interface has a parent scheduler.

40. (New) The method according to Claim 29, wherein at least one of the schedulables is a scheduler comprising a hierarchy of schedulables, such that activation of the schedulable scheduler comprises an activation that propagates up the hierarchy of schedulables.

41. (New) In a system-level design and integration system which facilitates composing behavior, capturing architecture and mapping behavior onto architecture, a method for specification of scheduling for simulation and implementation of consumer embedded systems; the method comprising the steps of:

a) identifying the schedulers in an architecture;

b) identifying the schedulables constituting behavior;

c) assigning schedulables and schedulers to each other;

d) arranging for the schedulers to find their respective assigned schedulables;

e) arranging for the schedulables to find their respective assigned scheduler;

f) sending an event to a schedulable;

g) sending an activation notice from the schedulable in step f) to its assigned scheduler;

h) sending a message from the assigned scheduler of step g) to the schedulable of step f) to start its behavior's reaction;

h1) having the scheduler of step g) determine whether the reaction of the schedulable of step f) must be preempted before it is finished;

h2) if in step h1) preemption is required, sending a suspend message from the scheduler of step g) to the schedulable of step f) to temporarily halt the execution of its reaction;

h3) if in step h1) preemption is required, causing the schedulable of step f) to halt its reaction in response to the suspend message;

h4) if in step H1) preemption is required, eventually sending a resume message from the scheduler of step g) to the schedulable of step f); and

i) sending a finish notice from the schedulable of step f) to the scheduler of step g) when said reaction is completed;

wherein:

each message comprises one of a series of schedulable-scheduler messages comprising a set of messages comprising: (1) an activation message from the schedulable to a scheduler indicating the schedulable wants to react to an event; (2) a start message from the scheduler instructing the schedulable to start; (3) a suspend message sent form the scheduler to suspend the schedulable's reaction; (4) a resume message from the scheduler to the

schedulable; and (5) a completion message from the schedulable to the scheduler;

the messages are modeled based on a stored value of how long it takes to transition through states specified by the message;

the schedulables react to events, and the events comprise a set of events having a timestamp that indicates an order in which a simulator processes the events;

the schedulables are activated based on a combination of non fixed frequency activation of the schedulable and a priority assigned to the schedulable;

resources utilized to process the schedulables are architectural resources allocated to the schedulables via a contention process;

at least one schedulable is a parent scheduler comprising a hierarchy of lesser schedulables including a scheduler configured to implement a scheduler interface; and

activation of the parent schedulable comprises an activation that propagates up the hierarchy of schedulables.

42. (New) The method recited in Claim 6, wherein the assigned transition times are assigned based on a set of functions dependent upon a state of the scheduler.